

DESIGN PATTERN

Membuat Proses CRUD data Customer Pada Aplikasi Laundry

Dosen Pengampu :

Afdhal Dinilhak. M.Kom



OLEH :

SHERLY SUKMADIRA PUTRI

2311532015

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2024

BAB I

PENDAHULUAN

A. Latar Belakang

Design Patterns adalah solusi standar yang telah terbukti (teruji) untuk menyelesaikan masalah umum yang sering terjadi dalam pengembangan perangkat lunak. Design Patterns bukanlah hal yang bisa ditranslate ke dalam kode program secara langsung, melainkan hanya template atau panduan cara menyelesaikan masalah. Terdapat banyak sekali desain pattern, dan untuk implementasi ke dalam kode programnya mungkin akan sedikit berbeda tergantung teknologi dan bahasa pemrograman yang digunakan. Dengan memahami design pattern kita akan lebih mudah dalam mengatasi masalah yang terjadi dalam pembuatan aplikasi/software. Karena kita sudah tahu solusi-solusi umum yang harus dilakukan, berbeda jika belum mengerti design pattern maka kita mencari solusi dari permasalahan yang dihadapi dengan mencobacoba solusi versi kita sendiri padahal sebenarnya sudah ada solusi yang sudah terbukti untuk masalah tersebut.

B. Tujuan

Mampu memahami dan menggunakan beberapa jenis design pattern pada kasus CRUD sebuah aplikasi.

BAB II

ALAT

1. Comp MySQL / XAMPP
2. MySQL / XAMPP
3. MySQL connector atau Connector/J

BAB III

LANDASAN TEORI

A. Xampp

XAMPP yaitu paket software yang terdiri dari Apache HTTP Server, MySQL, PHP dan Perl yang bersifat open source, xampp biasanya digunakan sebagai development environment dalam pengembangan aplikasi berbasis web secara localhost.

Apache berfungsi sebagai web server yang digunakan untuk menjalankan halaman web, MySQL digunakan untuk manajemen basis data dalam melakukan manipulasi data, PHP digunakan sebagai bahasa pemrograman untuk membuat aplikasi berbasis web

B. Design Pattern

metode yang dibuat untuk membantu tim pengembang dalam menemukan solusi dari masalah-masalah umum yang muncul saat pengembangan perangkat lunak sedang berlangsung.

C. MySQL

MySQL adalah sebuah relational database management system (RDBMS) open-source yang digunakan dalam pengelolaan database suatu aplikasi, MySQL ini dapat digunakan untuk menyimpan, mengelola dan mengambil data dalam format table

D. MySQL Connector/J

MySQL Connector/J adalah driver yang digunakan untuk menghubungkan aplikasi berbasis java dengan database MySQL sehingga dapat berinteraksi seperti menyimpan, mengubah, mengambil dan menghapus data.

Beberapa fungsi MySQL connector yaitu :

- Membuka koneksi ke database MySQL
- Mengirimkan permintaan SQL ke server MySQL
- Menerima hasil dari permintaan SQL

- Menutup koneksi ke database MySQL

E. DAO (Data Access Object)

DAO (Data Access Object) merupakan object yang menyediakan abstract interface terhadap beberapa method yang berhubungan dengan database seperti mengambil data (read), menyimpan data(create), menghapus data (delete), mengubah data(update).

Tujuan penggunaan DAO yaitu :

- Meningkatkan modularitas yaitu memisahkan logika akses data dengan logika bisnis sehingga memudahkan untuk dikelola
- Meningkatkan reusabilitas yaitu DAO dapat digunakan Kembali
- Perubahan pada logika akses data dapat dilakukan tanpa mempengaruhi logika bisnis.

F. Interface

Interface dalam Bahasa java yaitu mendefinisikan beberapa method abstrak yang harus diimplementasikan oleh class yang akan menggunakannya.

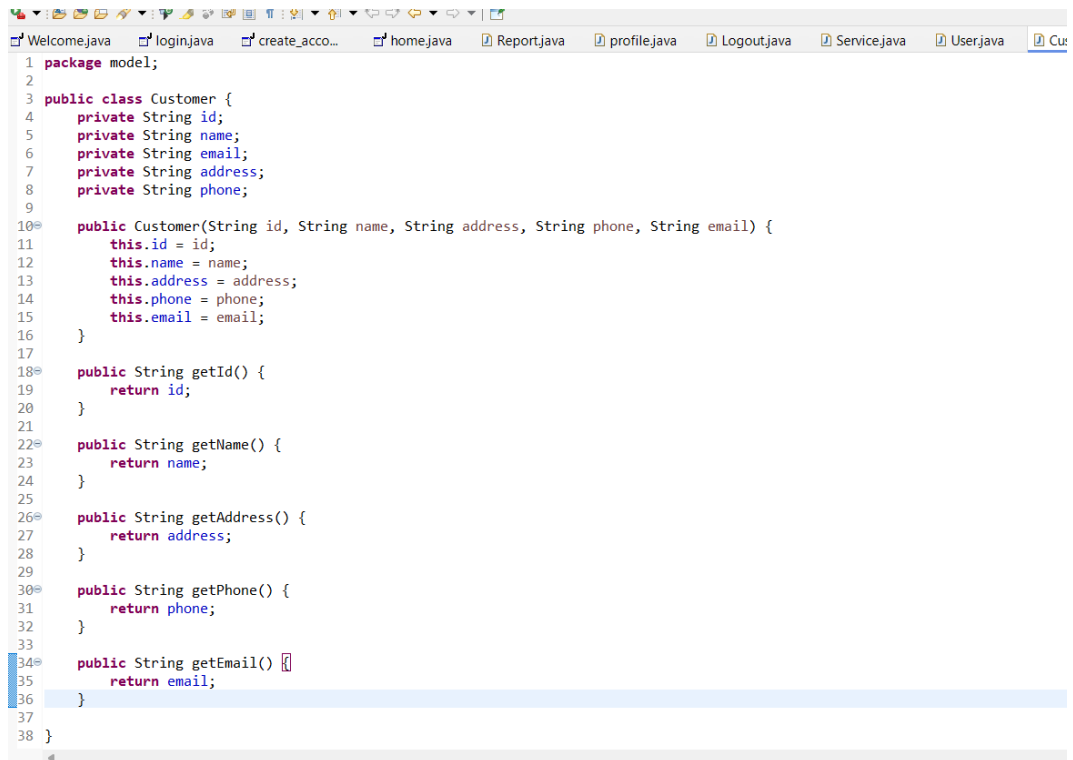
G. CRUD

CRUD (Create, Read, Update, Delete) merupakan fungsi dasar atau umum yang ada pada sebuah aplikasi yang mana fungsi ini dapat membuat, membaca, mengubah dan menghapus suatu data pada database aplikasi.

BAB IV

PROSEDUR DAN PENGAPLIKASIAN

1. Buat class Customer pada package model serta tambahkan method getter dan setter



```

1 package model;
2
3 public class Customer {
4     private String id;
5     private String name;
6     private String email;
7     private String address;
8     private String phone;
9
10    public Customer(String id, String name, String address, String phone, String email) {
11        this.id = id;
12        this.name = name;
13        this.address = address;
14        this.phone = phone;
15        this.email = email;
16    }
17
18    public String getId() {
19        return id;
20    }
21
22    public String getName() {
23        return name;
24    }
25
26    public String getAddress() {
27        return address;
28    }
29
30    public String getPhone() {
31        return phone;
32    }
33
34    public String getEmail() {
35        return email;
36    }
37 }

```

2. Buat sebuah class baru dengan nama CustomerBuilder di dalam package model. Class CustomerBuilder merupakan builder untuk membuat objek Customer. Dengan menggunakan Builder, kita dapat dengan mudah membuat objek yang memiliki kombinasi atribut yang berbeda tanpa harus membuat banyak constructor. Selanjutnya, copy semua atribut dari yang ada di class Customer ke dalam class CustomerBuilder dan buat method setter untuk masing-masing atribut tersebut. Di sini kita bisa memberikan default value untuk atribut yang ingin diberikan default value. Kemudian, buat sebuah method dengan nama build dengan return value adalah Customer.

```

Welcome.java login.java create_acco... home.java Report.java profile.java Logout.java Service.java
1 package model;
2
3 public class CustomerBuilder {
4     private String id;
5     private String name;
6     private String address;
7     private String phone;
8     private String email;
9
10
11 public CustomerBuilder() {
12
13 }
14
15 public CustomerBuilder setId(String id) {
16     this.id = id;
17     return this;
18
19 public CustomerBuilder setName(String name) {
20     this.name = name;
21     return this;
22
23 public CustomerBuilder setAddress(String address) {
24     this.address = address;
25     return this;
26
27 public CustomerBuilder setPhone(String phone) {
28     this.phone = phone;
29     return this;
30
31 public CustomerBuilder setEmail(String email) {
32     this.email = email;
33     return this;
34
35 public Customer build() {
36     return new Customer(id, name, address, phone, email);
37 }
38

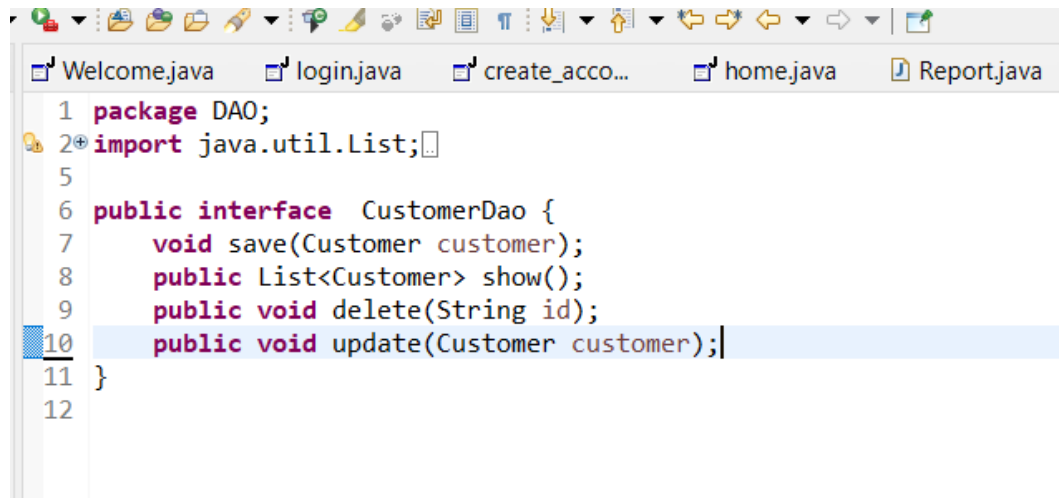
```

3. Buat tampilan CRUD pada JFrame di dalam package ui dengan nama CustomerFrame

The screenshot shows a Java Swing window titled "CustomerFrame" with a light gray background. It contains a form with four text input fields labeled "Name", "Address", "Phone", and "Email". Below the form are four buttons: "Save" (green border), "Update" (blue border), "Delete" (red border), and "Cancel" (yellow border). At the bottom of the window is a table with five columns: "ID", "Name", "Address", "Phone", and "Email". The table has four empty rows for data entry.

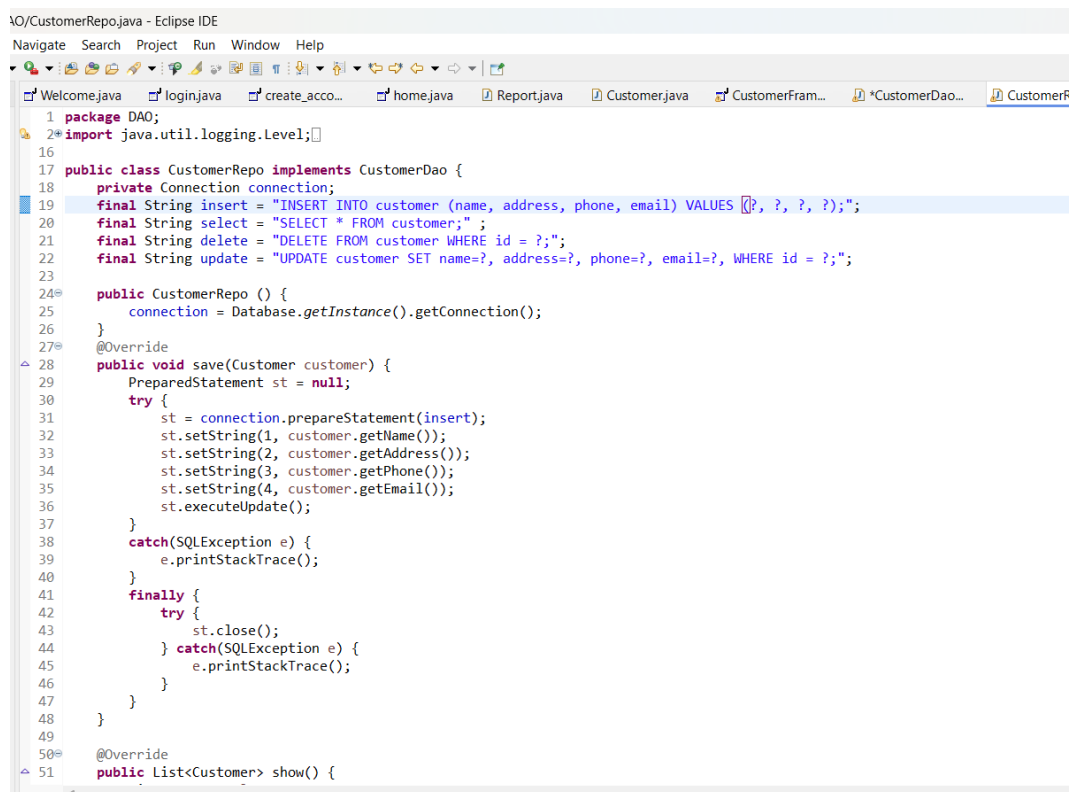
ID	Name	Address	Phone	Email

4. Buat class CustomerDao pada package dao



```
1 package DAO;
2 import java.util.List;
3
4
5
6 public interface CustomerDao {
7     void save(Customer customer);
8     public List<Customer> show();
9     public void delete(String id);
10    public void update(Customer customer);
11 }
12
```

5. Jangan lupa untuk bikin CustomerRepo juga untuk di dalam package dao untuk mengimplementasikan CustomerDao. Implementasikan masing-masing method yang terdapat pada interface CustomerDao yaitunya method save, update, delete, dan show.



```
1 package DAO;
2 import java.util.logging.Level;
3
4
5
6 public class CustomerRepo implements CustomerDao {
7     private Connection connection;
8     final String insert = "INSERT INTO customer (name, address, phone, email) VALUES (?, ?, ?, ?)";
9     final String select = "SELECT * FROM customer";
10    final String delete = "DELETE FROM customer WHERE id = ?";
11    final String update = "UPDATE customer SET name=?, address=?, phone=?, email=?, WHERE id = ?";
12
13    public CustomerRepo () {
14        connection = Database.getInstance().getConnection();
15    }
16
17    @Override
18    public void save(Customer customer) {
19        PreparedStatement st = null;
20        try {
21            st = connection.prepareStatement(insert);
22            st.setString(1, customer.getName());
23            st.setString(2, customer.getAddress());
24            st.setString(3, customer.getPhone());
25            st.setString(4, customer.getEmail());
26            st.executeUpdate();
27        }
28        catch(SQLException e) {
29            e.printStackTrace();
30        }
31        finally {
32            try {
33                st.close();
34            } catch(SQLException e) {
35                e.printStackTrace();
36            }
37        }
38    }
39
40    @Override
41    public List<Customer> show() {
42    }
43
44    }
45
46    }
47
48    }
49
50    }
51
```

```

Velcome.java  login.java  create_acco...  home.java  Report.java  Customer.java  CustomerFram...  *C
0 @Override
1 public List<Customer> show() {
2     List<Customer> ls = null;
3     try {
4         ls = new ArrayList<Customer>();
5         Statement st = connection.createStatement();
6         ResultSet rs = st.executeQuery(select);
7         while(rs.next()) {
8             Customer customer = new CustomerBuilder()
9                 .setId(rs.getString("id"))
10                .setName(rs.getString("name"))
11                .setAddress(rs.getString("address"))
12                .setPhone(rs.getString("phone"))
13                .setEmail(rs.getString("email"))
14                .build(); // Pastikan ada metode build() yang mengembalikan objek Customer
15            ls.add(customer);
16        }
17    } catch (SQLException e) {
18        Logger.getLogger(UserDao.class.getName()).log(Level.SEVERE, null, e);
19    }
20    return ls;
21 }
2
3 @Override
4 public void delete(String id) {
5     PreparedStatement st = null;
6     try {
7         st = connection.prepareStatement(delete);
8         st.setString(1, id);
9         st.executeUpdate();
10    } catch (SQLException e) {
11        e.printStackTrace();
12    }
13    finally {
14        try {
15            st.close();
16        }
17    }

```



```

Welcome.java login.java create_acco... home.java Report.java Customer.java Custo
78         st.setString(1,id);
79         st.executeUpdate();
80
81     }catch(SQLException e) {
82         e.printStackTrace();
83
84     }finally {
85         try {
86             st.close();
87
88         }catch(SQLException e) {
89             e.printStackTrace();
90         }
91     }
92 }
93
94 @Override
95 public void update(Customer customer) {
96     PreparedStatement st = null;
97     try {
98         st = connection.prepareStatement(update);
99         st.setString(1, customer.getName());
100        st.setString(2, customer.getAddress());
101        st.setString(3, customer.getPhone());
102        st.setString(4, customer.getId());
103        st.executeUpdate();
104
105    }catch(SQLException e){
106        e.printStackTrace();
107
108    }
109    finally {
110        try {
111            st.close();
112        }catch(SQLException e) {
113            e.printStackTrace();
114        }
115    }

```

6. Untuk menampilkan data customer dalam bentuk table pada CustomerFrame kita perlu membuat Class baru dengan nama TableCustomer pada package table. TableCustomer ini akan meng-extends class AbstractTableModel yang merupakan class bawaan dari Java.

```
avigate Search Project Run Window Help
Welcome.java login.java create_acc... home.java Customer.java CustomerFram... TableCust...

1 package table;
2 import java.util.List;
3
4
5
6 public class TableCustomer extends AbstractTableModel{
7     List<Customer> ls;
8     private String[] columnNames = {"ID", "Name", "Address", "Phone", "Email"};
9     public TableCustomer(List <Customer> ls) {
10         this.ls = ls;
11     }
12     public int getRowCount() {
13         // TODO Auto-generated method stub
14         return ls.size();
15     }
16     public int getColumnCount() {
17         // TODO Auto-generated method stub
18         return 5;
19     }
20     public String getColumnName (int column) {
21         return columnNames[column];
22     }
23     public Object getValueAt(int rowIndex, int columnIndex) {
24         switch (columnIndex) {
25             case 0:
26                 return ls.get(rowIndex).getId();
27             case 1 :
28                 return ls.get(rowIndex).getName();
29             case 2 :
30                 return ls.get(rowIndex).getAddress();
31             case 3 :
32                 return ls.get(rowIndex).getPhone();
33             case 4 :
34                 return ls.get(rowIndex).getEmail();
35             default :
36                 return null;
37         }
38     }
39 }
40
```

7. Pada CustomerFrame, tambahkan instance id, ls yang akan menampung data List Customer, dan objek customerRepo. Selain itu, kita juga perlu membuat method untuk load data customer dan memasukkannya ke TableCustomer, serta method untuk me-reset form input customer

```

Welcome.java login.java create_acc... home.java Customer.java CustomerFrame... x TableCustom... *Cus
1 package Ui;
2
3 import java.awt.EventQueue;
31
32 public class CustomerFrame extends JFrame {
33     private static final long serialVersionUID = 1L;
34     private JPanel contentPane;
35     private JTextField txtName;
36     private JTextField txtAddress;
37     private JTextField txtPhone;
38     private JTextField txtEmail;
39     private JTable tableCustomer;
40
41
42     public static void main(String[] args) {
43         EventQueue.invokeLater(new Runnable() {
44             public void run() {
45                 try {
46                     CustomerFrame frame = new CustomerFrame();
47                     frame.setVisible(true);
48                     frame.loadTable();
49                 } catch (Exception e) {
50                     e.printStackTrace();
51                 }
52             }
53         });
54     }
55
56     //method reset
57     public void reset () {
58         txtName.setText("");
59         txtAddress.setText("");
60         txtPhone.setText(" ");
61         txtEmail.setText(" ");
62     }
63     //instance pada userframe
64     CustomerRepo cus = new CustomerRepo();

```

```

}
//instance pada userframe
CustomerRepo cus = new CustomerRepo();
List<Customer> ls;
public String id;

public void loadTable() {
    ls = cus.show();
    TableCustomer tc = new TableCustomer(ls);
    tableCustomer.setModel(tc);
    tableCustomer.getTableHeader().setVisible(true);
}

```

8. Tambahkan kode berikut pada button save agar data dapat disimpan ke database

ie.java - Eclipse IDE

arch Project Run Window Help



java login.java create_acco... home.java Customer.java CustomerFram... X

```
JScrollPane.setViewportViewView(tableCustomer);

JButton btnSave = new JButton("Save");
btnSave.setBackground(new Color(128, 255, 0));
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        CustomerBuilder customerBuild = new CustomerBuilder();
        customerBuild.setName(txtName.getText());
        customerBuild.setAddress(txtAddress.getText());
        customerBuild.setPhone(txtPhone.getText());
        customerBuild.setEmail(txtEmail.getText());

        customerBuild.setId(id);
        Customer customer = customerBuild.build();
        cus.save(customer);
        reset();
        loadTable();
        //Customer customer = new Customer();
        //customer.setName(txtName.getText());
        //customer.setAddress(txtAddress.getText());
        //customer.setPhone(txtPhone.getText());
        //cus.save(customer);
        //reset();
        //loadTable();
    }
});
```

BAB V
DOKUMENTASI HASIL

Name	<input type="text" value="h"/>
Address	<input type="text" value="b"/>
Phone	<input type="text" value="2"/>
Email	<input type="text" value="b"/>
<div><div>Save</div><div>Update</div><div>Delete</div><div>Cancel</div></div>	

ID	Name	Address	Phone	Email
1	h	b	2	b
2	j	ih	0	jv

—□×

Name

Address

Phone

Email

Save

Update

Delete

Cancel

ID	Name	Address	Phone	Email
1	h	b	2	b
2	j	ih	0	jv
3	h	b	2	b
4	bucel	komplek unand	823	shertydrp

BAB VI

SIMPULAN

Praktikum ini memberikan pemahaman yang mendalam mengenai pengembangan aplikasi menggunakan sistem Design Pattern yang digunakan untuk menyelesaikan berbagai permasalahan umum yang terjadi dalam pengembangan perangkat lunak. Design pattern memiliki beberapa manfaat diantaranya membantu mengurangi pengulangan logika, membuat kode lebih mudah dipahami dan diperbaiki ketika terjadi perubahan, dan membantu menyelesaikan masalah lebih cepat.